1. **(original)** A finite state model-based testing system comprising:

a model generation engine to generate a model of a software application to be tested; and

a graphical user interface to enable user entry of parameters for defining the model.

2. **(original)** A finite state model-based testing system as recited in claim 1, wherein the user interface enables a user to enter state information and transition information about the software application, the transition information describing a next state of the software application after an input has been applied to a current state of the software application.

3. **(original)** A finite state model-based testing system as recited in claim 1, wherein the user interface enables a user to enter state information about the software application, the state information comprising:

an operational mode of the software application, wherein the operational mode is an attribute of a particular state of the software application;

at least one modal value associated with the operational mode, wherein the modal value describes a behavior of the operational mode; and

an input of the software application.

4. (original) A finite state model-based testing system as recited in claim 1, wherein the user interface enables a user to enter transition information about the software application, the transition information comprising:

a current state of the software application, the current state being associated with an input of the software application; and

a next state of the software application, the next state indicating the state of the software application after the input has been applied to the current state of the software application.

5. (original) A finite state model-based testing system as recited in claim 1, wherein the user interface comprises a model editor to enable user entry of an operational mode, a modal value associated with the operational mode, and an input of the software application for defining the model.

6. (original) A finite state model-based testing system as recited in claim 1, wherein the user interface comprises a rules editor to enable user entry of an input of the software application, a current state of the software application, and a next state of the software application indicating the state of the software application after the input has been applied to the current state of the software application for defining the model.

7. (original) A finite state model-based testing system as recited in claim 1, wherein the model of the software application is a state table, the state table having at least one state table entry, and wherein:

a state table entry comprises:

(1) a current state of the software application;

(2) an input of the software application;

(3) a next state of the software application, the next state indicating the state of the software application after the input has been applied to the current state of the software application;

the model generation engine evaluates the current state of the software application to determine if an input of the software application can be applied to the current state and in the event that the input can be applied to the current state, writes a state table entry out to the state table.

8. (original) A finite state model-based testing system as recited in claim 1, wherein the user interface comprises a graph traversal menu to enable a user to select a graph traversal program and generate a test sequence of inputs for the software application.

9. (original) A finite state model-based testing system as recited in claim 1, further comprising a graph traversal program to generate a test sequence of inputs for the software application, the test sequence of inputs generated from the model of the software application with the graph traversal program.

10. (original) A finite state model-based testing system as recited in claim 1, wherein the user interface comprises a test execution menu to enable a user to select a test driver program and initiate a test of the software application.

11. (original) A finite state model-based testing system as recited in claim 1, further comprising a test driver program to execute a test sequence of application inputs on the software application.

12. (original) A user interface for testing a software application, the user interface comprising:

a model editor to enable user entry of state information to define a model of a software application to be tested; and

a rules editor to enable user entry of transition information to further define the model of the software application to be tested, the transition information describing a next state of the software application after an input has been applied to a current state of the software application.

13. (original) A user interface as recited in claim 12, wherein the user interface is a graphical user interface.

14. (original) A user interface as recited in claim 12, wherein the state information comprises:

an operational mode of the software application, wherein the operational mode is an attribute of a particular state of the software application;

at least one modal value associated with the operational mode, wherein the modal value describes a behavior of the operational mode; and

an input of the software application.

15. (original) A user interface as recited in claim 12, wherein the transition information comprises:

a current state of the software application, the current state being associated with an input of the software application; and

a next state of the software application, the next state indicating the state of the software application after the input has been applied to the current state of the software application.

16. (original) A user interface as recited in claim 12, wherein the model editor comprises:

an operational modes entry field to enable user entry of an operational mode of the software application; and

an operational modes list field to display the operational mode.

17. (original) A user interface as recited in claim 12, wherein the model editor comprises:

a modal values entry field to enable user entry of a modal value associated with an operational mode of the software application; and

a modal values list field to display the modal value.

18. (original) A user interface as recited in claim 12, wherein the model editor comprises:

an inputs entry field to enable user entry of an input of the software application; and

an inputs list field to display the input of the software application.

19. (original) A user interface as recited in claim 12, wherein the rules editor comprises fields to display the state information that can be entered using the model editor, the fields comprising:

inputs fields to display inputs of the software application, wherein the inputs fields also enable user selection of an input of the software application;

operational modes fields to display operational modes of the software application, wherein the operational modes fields also enable user selection of an operational mode; and

modal values fields to display modal values associated with an operational mode, wherein the modal values fields also enable user selection of a modal value.

20. (original) A user interface as recited in claim 19, wherein the rules editor enables user entry of the transition information, and wherein:

the transition information comprises:

a current state of the software application, the current state being associated with the input of the software application;

a next state of the software application, the next state indicating the state of the software application after the input has been applied to the current state of the software application;

the rules editor comprises:

an inputs field to enable user entry of the input;

a current state operational mode field to enable user entry of the operational mode as a current state operational mode;

a current state modal value field to enable user entry of the modal value associated with the current state operational mode;

a next state operational mode field to enable user entry of the operational mode as a next state operational mode; and

a next state modal value field to enable user entry of the modal value associated with the next state operational mode.

21. (original) A user interface as recited in claim 12, wherein the model is a state table having at least one state table entry, the state table entry having a current state of the software application, an input of the software application, and a next state of the software application, the next state indicating the state of the software application after the input has been applied to the current state of the software application; and

the model editor enables user initiation of a model generation engine to generate the model of the software application, the model generation engine being configured to evaluate a current state of the software application to determine if an input of the software application can be applied to the current state and in the event that the input can be applied to the current state, writes a state table entry out to the state table.

22. (original) A user interface as recited in claim 12, wherein the model editor enables user initiation of a graph traversal program to generate a test sequence of inputs for the software application.

23. (original) A user interface as recited in claim 12, wherein the user interface further comprises a graph traversal menu to enable user selection of a graph traversal program to generate a test sequence of inputs for the software application.

24. (original) A user interface as recited in claim 23, wherein the graph traversal menu comprises:

a graph traversal program field to enable user selection of the graph traversal program;

a model file field to enable user selection of the model of the software application to be tested; and

a test sequence file field to enable user entry of a memory storage location for a test sequence file, the test sequence file containing the test sequence of inputs for the software application.

25. (original) A user interface as recited in claim 12, wherein the model editor enables user initiation of a test driver program to read a test sequence of inputs for the software application and apply the test sequence to the software application.

26. (original) A user interface as recited in claim 12, wherein the user interface further comprises a test execution menu to enable user selection of a test driver program to read a test sequence of inputs for the software application and apply the test sequence to the software application.

27. (original) A user interface as recited in claim 26, wherein the test execution menu comprises:

a test driver program field to enable user selection of the test driver program;

a test sequence file field to enable user selection of a test sequence file containing the test sequence of inputs for the software application to be tested; and

a test monitoring interval field to enable user entry of a timing interval to define how often the test driver program can be monitored to detect a failure of the test driver program.

28. (original) A user interface to enable user entry of parameters to define a model of a software application to be tested, the user interface comprising:

an operational modes field to enable user entry of an operational mode of the software application, the operational mode being an attribute of a particular state of the software application; and

a modal values field to enable user entry of at least one modal value associated with the operational mode, the modal value describing a behavior of the operational mode.

29. (original) A user interface as recited in claim 28, further comprising an input field to enable user entry of an input of the software application.

30. (original) A user interface as recited in claim 29, further comprising:

an operational modes list field to display the operational mode;

a modal values list field to display the modal value; and

an inputs list field to display the input of the software application.

31. (original) A user interface as recited in claim 28, wherein the user interface enables user initiation of a graph traversal program to generate a test sequence of inputs for the software application.

32. (original) A user interface as recited in claim 28, wherein the user interface enables user initiation of a test driver program to read a test sequence of inputs for the software application and apply the test sequence to the software application.

33. (original) A user interface to enable user entry of parameters to define a model of a software application to be tested, the user interface comprising:

an inputs field to enable user entry of an input of the software application;

current state fields to enable user entry of a current state of the software application, the current state being associated with the input; and

next state fields to enable user entry of a next state of the software application, the next state indicating the state of the software application after the input has been applied to the current state of the software application.

34. (original) A user interface as recited in claim 33, further comprising a rules field to enable user entry of a rule to describe a transition of the software application from a current state to a next state.

35. (original) A user interface as recited in claim 34, further comprising a control to enable a user to disable a rule such that the model of the software application will be defined without the rule.

36. (original) A user interface as recited in claim 33, wherein:

the current state fields include:

a current state operational mode field to enable user entry of a current state operational mode of the software application;

a current state modal value field to enable user entry of at least one current state modal value associated with the current state operational mode;

the next state fields include:

a next state operational mode field to enable user entry of a next state operational mode of the software application; and

a next state modal value field to enable user entry of at least one next state modal value associated with the next state operational mode.

37. (original) A user interface as recited in claim 36, wherein:

the current state fields include:

a current state relational operator field to indicate the current state modal value relation to the current state operational mode;

a current state concatenation operator field to indicate the relation between a first current state rule criteria and a second current state rule criteria.

the next state fields include:

a next state relational operator field to indicate the next state modal value relation to the next state operational mode; and

a next state concatenation operator field to indicate the relation between a first next state rule criteria and a second next state rule criteria.

38. (original) A data structure stored on a computer readable medium comprising:

at least one mode data structure to hold an operational mode of a software application to be tested and at least one modal value associated with the operational mode; and

at least one rule data structure to hold an input of the software application to be tested, a current state of a software application, and a next state of the software application.

39. (original) A data structure as recited in claim 38, wherein the mode data structure is a linked list of one or more mode data structures.

40. (original) A data structure as recited in claim 38, wherein the rule data structure is a linked list of one or more rule data structures.

41. (original) A data structure as recited in claim 38, wherein:

the current state and the next state are defined by rule criteria data structures; and

the rule criteria data structures are stored in a linked list.

42. (original) A finite state model-based testing system comprising:

a model editor to enable user entry of state information to define a model of a software application to be tested;

a rules editor to enable user entry of transition information to further define the model of the software application, the transition information describing a next state of the software application after an input has been applied to a current state of the software application;

a model generation engine to generate the model of the software application;

a graph traversal menu to enable user selection of a graph traversal program to generate a test sequence of inputs for the software application; and

a test execution menu to enable user selection of a test driver program to read the test sequence of inputs for the software application and apply the test sequence to the software application.

43. **(original)** A finite state model-based testing system as recited in claim 42, wherein the model editor comprises:

operational modes fields to enable user entry of an operational mode of the software application;

modal values fields to enable user entry of a modal value associated with the operational mode; and

inputs fields to enable user entry of an input of the software application.

44. **(original)** A finite state model-based testing system as recited in claim 42, wherein the rules editor comprises fields to display the state information that can be entered using the model editor, the fields comprising:

inputs fields to display inputs of the software application, wherein the inputs fields also enable user selection of an input of the software application;

operational modes fields to display operational modes of the software application, wherein the operational modes fields also enable user selection of an operational mode; and

modal values fields to display modal values associated with an operational mode, wherein the modal values fields also enable user selection of a modal value.

45. (original) A finite state model-based testing system as recited in claim 42, wherein the model is a state table having at least one state table entry, the state table entry having a current state of the software application, an input of the software application, and a next state of the software application; and

the model editor enables user initiation of the model generation engine which is configured to evaluate a current state of the software application to determine if an input of the software application can be applied to the current state and in the event that the input can be applied to the current state, writes a state table entry out to the state table.

46. (original) A finite state model-based testing system as recited in claim 42, wherein the model editor facilitates user initiation of the rules editor.

47. (original) A finite state model-based testing system as recited in claim 42, wherein the model editor facilitates user initiation of the graph traversal menu.

48. (original) A finite state model-based testing system as recited in claim 42, wherein the model editor facilitates user initiation of the test execution menu.

49. (original) A computer system comprising:

a processor;

a memory;

a user interface application stored in the memory and executable on the processor to facilitate user definition of a finite-state model to test a software application;

the user interface application having computer readable instructions to display a graphical user interface; and

a model generation engine stored in memory and executable on the processor to generate the model of the software application.

50. (original) A computer system as recited in claim 49, wherein the user interface enables a user to enter state information and transition information about the software application, the transition information describing a next state of the software application after an input has been applied to a current state of the software application.

51. (original) A computer system as recited in claim 49, wherein the user interface comprises a model editor to enable user entry of operational modes, modal values, and inputs of the software application to define the model.

52. (original) A computer system as recited in claim 49, wherein the user interface comprises a rules editor to enable user entry of an input of the software application, a current state of the software application, and a next state of the software application to define the model.

53. (original) A computer system as recited in claim 49, further comprising at least one graph traversal program stored in the memory and executable on the processor to generate a test sequence of inputs for the software application, the user interface presenting a graph traversal menu to enable a user to select the graph traversal program.

54. (original) A computer system as recited in claim 49, further comprising at least one test driver program stored in the memory and executable on the processor to execute a test sequence of application inputs on the software application, the user interface presenting a test execution menu to enable a user to select the test driver program.

55. (original) A computer system as recited in claim 49, further comprising a data structure stored in the memory, the data structure comprising:

at least one mode data structure to hold an operational mode of a software application and at least one modal value associated with the operational mode; and

at least one rule data structure to hold an input of the software application, a current state of a software application, and a next state of the software application.

56. (original) A method comprising:

presenting a graphical user interface that facilitates user entry of state information and transition information about a software application to be tested; and

generating a model of the software application using the state information and the transition information.

57. (original) A method as recited in claim 56, further comprising:

presenting a graphical user interface that facilitates user selection of a graph traversal program; and

generating a test sequence of inputs for the software application.

58. (original) A method as recited in claim 56, further comprising:

presenting a graphical user interface that facilitates user selection of a test driver program; and

executing a test sequence of application inputs on the software application.

59. (original) A method as recited in claim 56, further comprising enabling a user to define a transition rule of the software application, wherein the enabling comprises presenting a graphical user interface to facilitate user entry of an input of the software application, a current state of the software application associated with the input, and a next state of the software application.

60. (original) A method as recited in claim 56, further comprising:

presenting a graphical user interface that facilitates a user defining a transition rule of the software application and disabling the transition rule; and

generating the model of the software application without incorporating the disabled transition rule.

61. (original) A method as recited in claim 56, wherein generating a model of the software application comprises:

evaluating a current state of the software application to determine if an input of the software application can be applied to the current state; and

in the event that the input can be applied to the current state, writing a state table entry out to a state table.

62. (original) A method as recited in claim 56, further comprising enabling a user to define the state information, wherein the enabling comprises presenting a graphical user interface to facilitate user entry of an operational mode of the software application, at least one modal value associated with the operational mode, and an input of the software application.

63. (original) A method as recited in claim 56, further comprising enabling a user to define the transition information, wherein the enabling comprises presenting a graphical user interface to facilitate user entry of a current state of the software application, the current state being associated with an input of the software application, and a next state of the software application, the next state indicating the state of the software application after the input has been applied to the current state of the software application.

64. (original) A computer-readable medium comprising computer executable instructions that, when executed, direct a computing system to perform the method of claim 56.

65. (original) A method comprising:

presenting a user interface that facilitates user entry of state information and transition information about a software application to be tested;

initiating, via the user interface, generation of a model of the software application;

selecting, via the user interface, a graph traversal program that generates a test sequence of inputs for the software application; and

selecting, via the user interface, a test driver program that executes a test sequence of application inputs on the software application.

66. (original) A computer-readable medium comprising computer executable instructions that, when executed, direct a computing system to perform the method of claim 65.

67. (original) A method comprising:

receiving state information about a software application to be tested;

receiving transition information about the software application;

generating a model of the software application;

from the model, generating a test sequence of inputs for the software application with a graph traversal program; and

executing a test sequence of application inputs on the software application.

68. (original) A method as recited in claim 67, wherein generating a model of the software application comprises:

evaluating a current state of the software application to determine if an input of the software application can be applied to the current state; and

in the event that the input can be applied to the current state, writing a state table entry out to a state table.

69. (original) A computer-readable medium comprising computer executable instructions that, when executed, direct a computing system to perform the method of claim 67.

70. (original) A computer-readable medium comprising computer executable instructions that, when executed, direct a computing system to:

receive state information about a software application to be tested;

receive transition information about the software application;

generate a model of the software application;

from the model, generate a test sequence of inputs for the software application with a graph traversal program; and

execute a test sequence of application inputs on the software application.